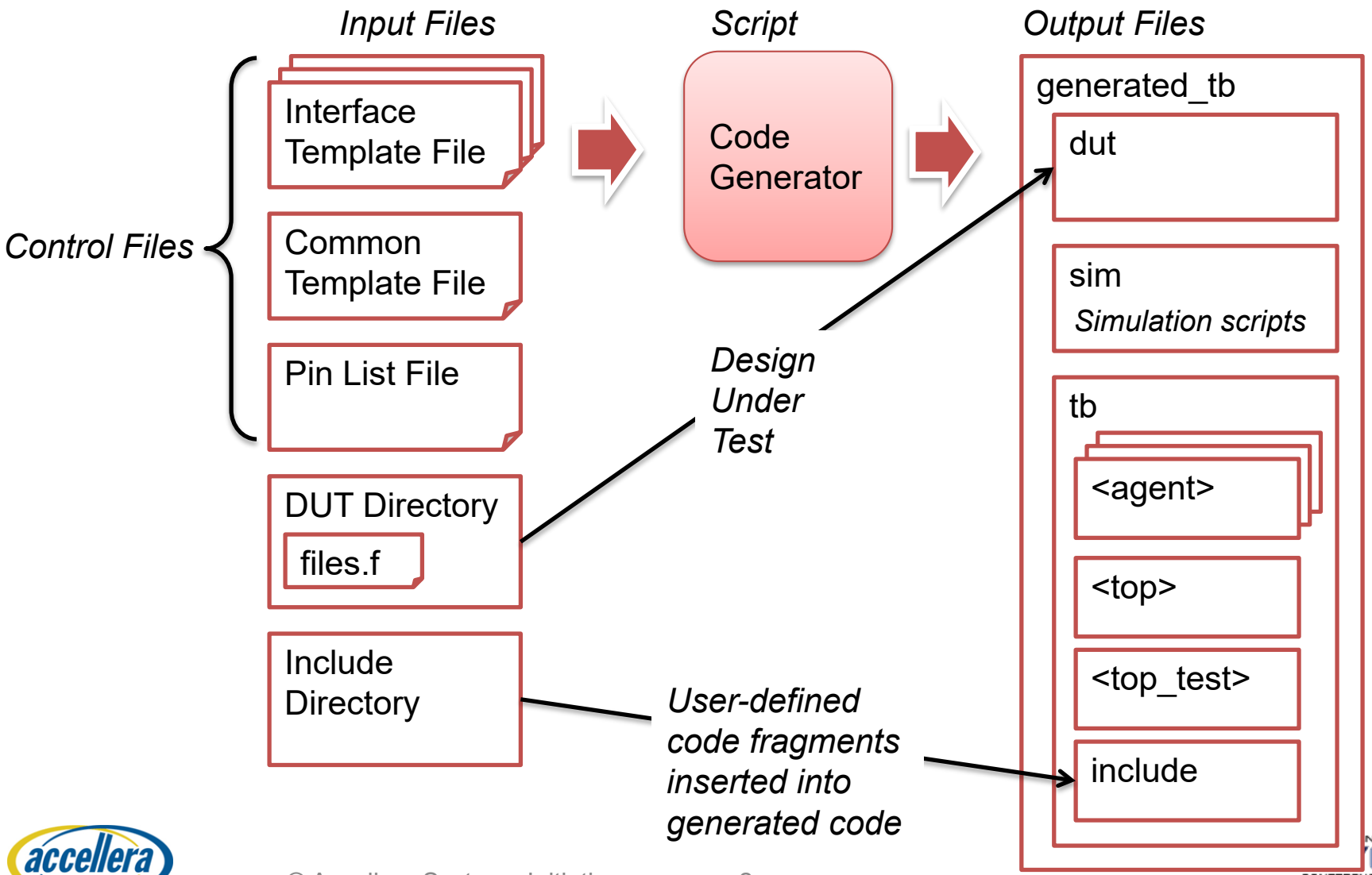
How Far Can You Take UVM Code Generation and Why Would You Want To?

John Aynsley, Doulos

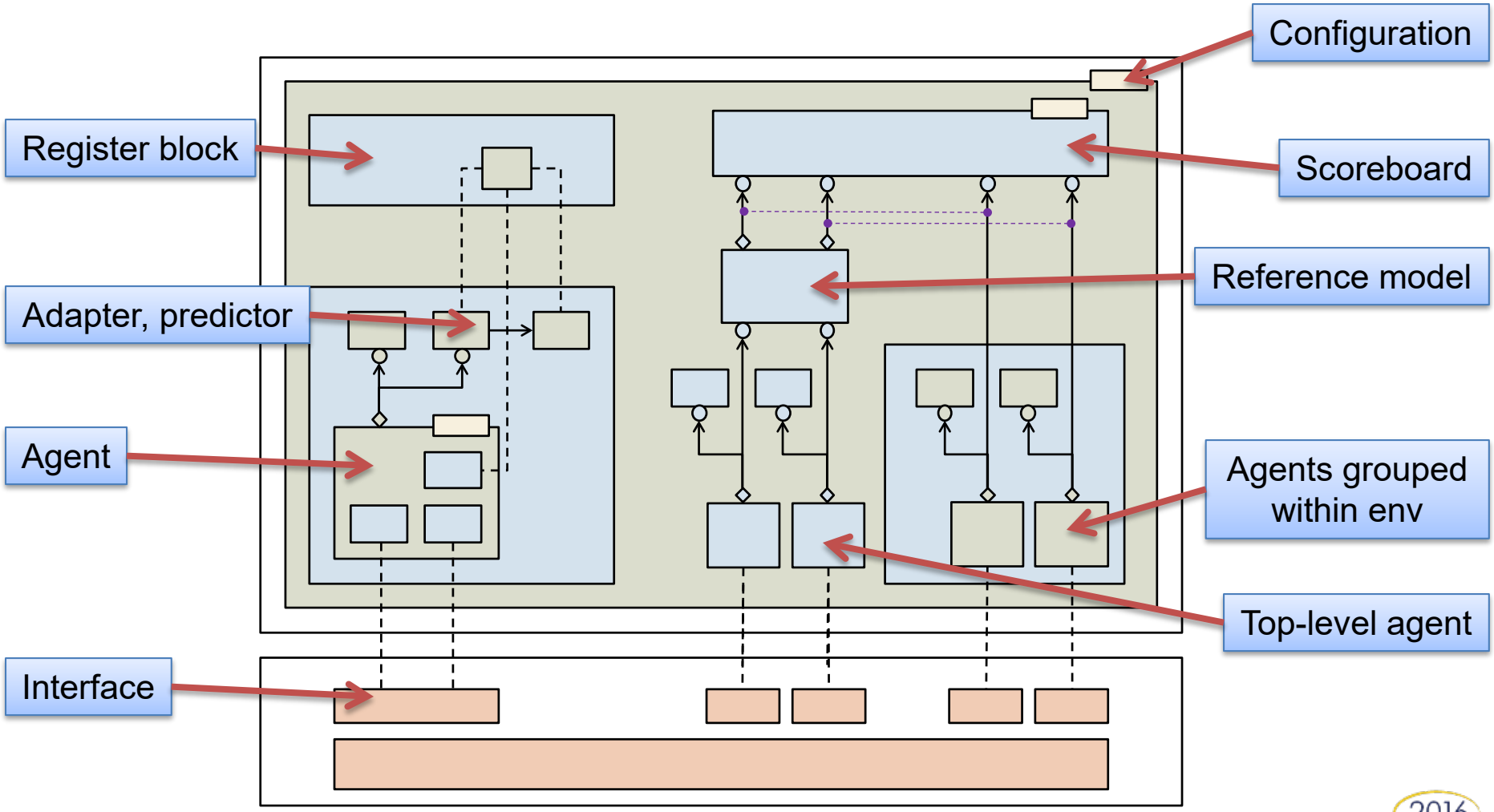
accellera SYSTEMS INITIATIVE

2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Code Generator Inputs & Outputs



Easier UVM Generator



Ways to Use a Code Generator

Generate examples as a learning aid

An initial framework for production code

Continually regenerate code throughout the project

Goals for Code Generation

The quality of the code

Better than I write

The quality of the generator

Generates what I want

Supports iteration

Regeneration & Flexibility

Must be able to regenerate the entire output

(lost or corrupted files, regression)

Insert user-defined code at predefined points

Suppress automatic generation

It still won't be enough!

How Far Can You Take UVM Code Generation? (1)

No limits! A few examples ...

Organising classes into packages using `include

Indentation and alignment – pretty printing

How to structure the complete UVM environment

Which UVM features to use where (best practice)

How Far Can You Take UVM Code Generation? (2)

Contributing to automation and productivity

Instantiating and connecting DUT and interfaces

Stitching together all the VIP at the top level

Generating methods for each transaction class

Generating simulation scripts

How Far Can You Take UVM Code Generation? (3)

Working example code

Default clock and reset generation

Default covergroups and sampling

Default end_of_elaboration to print diagnostics

Default sequences

Why Would You Want To?

For learners

Reinforce training with complete, working examples

For all users

Consistency

Productivity – automatic generation

Summary

Consistency and automation are worth having!

Get our generator from

http://www.doulos.com/easier

Example

http://www.edaplayground.com/x/65x